Модел. и анализ информ. систем. Т. **21**, № **5** (2014) 93–101

УДК 004.42

Применение методов оптимизации для выполнения молекулярного докинга на графических процессорах

Фарков М. А., Легалов А. И.

Сибирский Федеральный Университет
Россия, 660041, г. Красноярск, пр. Свободный, 79.

e-mail: mihail.farkov@gmail.com, legalov@mail.ru

получена 23 августа 2014

Ключевые слова: CUDA, GPGPU, молекулярный докинг, виртуальный скрининг, лиганд-белковый докинг, дифференциальная эволюция

В работе проведён анализ методов оптимизации для решения задачи молекулярного докинга. Сформированы дополнительные требования, предъявляемые к методам оптимизации при их реализации на графических процессорах. Выбран перспективный метод для графических процессоров и продемонстрирована его реализация.

1. Введение

Молекулярным лиганд-белковым докингом называют моделирование взаимодействия молекулы биомишени (белка) с небольшой молекулой (лигандом). Цель моделирования заключается в определении принципиальной возможности протекания химической реакции, оценки её характеристик и отбора наиболее перспективных соединений для дальнейших, более точных испытаний. Основной областью применения молекулярного лиганд-белкового докинга является разработка лекарств. В процессе моделирования взаимодействия лиганд различным образом позиционируется в активном сайте связывания молекулы биомишени, после чего выполняется оценка энергии взаимодействия [1]. Даже при достаточно ограниченном пространстве позиционирования лиганда и учёте 6-ти основных степеней свободы в трёхмерном пространстве (3 степени смещения и 3 степени вращения), количество операций позиционирования и оценки энергии велико. При этом молекула лиганда подвижна при взаимодействии с биомишенью. В частности, происходит вращение по торсионным углам. Для более точной оценки необходимо, помимо основных степеней свободы, учитывать вращения по торсионным углам, что, в зависимости от лиганда, может существенно увеличить количество степеней свободы, которые необходимо принимать во внимание при расчёте. В такой ситуации исчерпывающий поиск – неосуществим. Для снижения вычислительной сложности принято использовать различные численные методы оптимизации. Использование массивно

параллельных вычислительных систем, таких как графические процессоры, чрезвычайно важно для ускорения процедуры докинга, но при этом на данный момент существует лишь несколько работ на эту тему, которые используют уже существующие программные пакеты в качестве цели для ускорения, что ограничивает заранее определённым методом оптимизации. Подбор процедур оптимизации с учётом специфических аппаратно-программных требований не производился. В рамках данной работы рассматриваются формирование дополнительных требований к методам оптимизации с учётом специфики графических процессоров и разработка модели выполнения молекулярного лиганд-белкового докинга с использованием графических процессоров.

2. Подбор метода оптимизации

Процесс оптимизации, как правило, разделён на два основных шага: глобальный поиск и локальный поиск. В зависимости от преследуемых целей локального поиска в алгоритме может не быть. В существующих решениях для выполнения глобальной оптимизации используются: генетический алгоритм [2, 3, 4], метод Монте-Карло [5, 6], муравьиный алгоритм [7].

При реализации метода оптимизации на графических процессорах к нему предъявляются дополнительные требования помимо точности и сходимости:

- высокая степень параллелизма метода, направленная на повышение точности алгоритма, что необходимо для обеспечения максимальной загрузки мультипроцессоров графического процессора;
- максимальная линейность вычислительного процесса, что необходимо для уменьшения количества дивергентных варпов, наличие которых приводит к линеаризации вычислений на CUDA (Compute Unified Device Architecture) ядрах и существенному падению производительности;
- отсутствие синхронизации между потоками вычислений либо наличие только барьерной синхронизации в пределах блока CUDA нитей;
- возможность регулярного распределения памяти под вспомогательные структуры методы, что необходимо для группировки запросов к глобальной памяти графического процессора, что в свою очередь существенно увеличивает производительность;
- минимальное количество входных параметров и этапов метода, что продиктовано жёсткими требованиями по количеству используемых регистров каждой нитью графического процессора (превышение такого лимита приводит к использованию локальной памяти графического процессора и уменьшению производительности).

После анализа существующих подходов для реализации был выбран метод дифференциальной эволюции, предложенный в [8]. Приближенное решение в методе дифференциальной эволюции задаётся некоторым вектором (DE vector), по аналогии с генетическим алгоритмом. При этом для осуществления метода требуется

некоторый набор таких векторов (кластер N векторов), начальное состояние которых определяется случайным образом. На каждой последующей итерации метода (G+1) для i-го вектора кластера происходит формирование мутантного вектора согласно формуле:

$$Vm_{i(G+1)} = V_{r1(G)} + F(V_{r2(G)} - V_{r3(G)}),$$

где $Vm_{i(G+1)}$ – мутантный вектор для i-го вектора; $V_{r1(G)}, V_{r2(G)}, V_{r3(G)}$ – векторы предыдущей итерации метода, выбранные случайно; $F \in [0, 2]$ – константа метода. После формирования мутантного вектора из него формируется тестовый вектор. Для каждого элемента мутантного вектора случайным образом генерируется вероятность перехода в тестовый вектор. Если эта вероятность выше заданного порога, элемент мутантного вектора переходит в тестовый вектор. В противном случае для данного элемента выбирается значение из i-го вектора предыдущей (G) итерации метода. После формирования тестового вектора происходит пересчёт целевой функции. Если её значение для тестового вектора оказывается предпочтительнее (в рассматриваемом случае, при уменьшении значения функции), тестовый вектор замещает i-вектор. Операция итеративно повторяется для всех N векторов кластера заданное количество раз.

Метод отвечает требуемым критериям. В отличие от метода Монте-Карло он хорошо масштабируется на большое количество потоков и обладает значительно большей скоростью сходимости, что важно из-за большого объема вычислений при обработке лиганда. Метод, по аналогии с генетическим алгоритмом, не требователен к памяти под вспомогательные структуры. При этом метод дифференциальной эволюции имеет регулярную структуру распределения памяти, использует меньше входных параметров и менее вычислительно сложен. В работе [9] для генетического алгоритма показано, что при реализации на графических процессорах таких этапов, как отбор, кроссинговер и мутация, было получено незначительное ускорение или ускорение достигнуто не было. Кроме того, итерация метода дифференциальной эволюции является максимально линейной. Алгоритм требует только барьерной синхронизации между всем кластером векторов.

3. Реализация

В работе используется сеточный подход для вычисления энергии межмолекулярного взаимодействия (электростатического и Ван-дер-Ваальсового) атомов лиганда и биомишени. Вычисления сеток осуществляются на графическом процессоре. Общий подход и алгоритмы вычисления сеток силовых полей на графических процессорах изложены в работах [1, 12]. Основной целью разрабатываемого решения является быстрый скрининг набора лигандов, в свете чего выбрана двухуровневая декомпозиция задачи (рисунок 1). Первый уровень декомпозиции включает в себя параллельную обработку нескольких лигандов на отдельных мультипроцессорах графического процессора. Второй уровень – это параллельная декомпозиция метода дифференциальной эволюции для отдельного лиганда.

Декомпозиция первого уровня обусловлена тем, что каждый мультипроцессор имеет собственные устройства диспетчеризации. Это позволяет обрабатывать различные лиганды независимо. Помимо этого каждый мультипроцессор имеет доста-

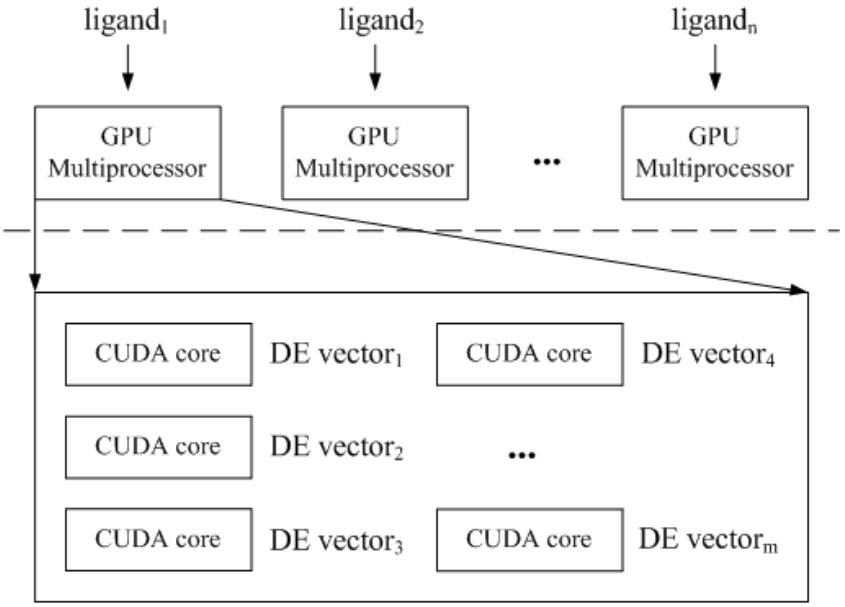

Рис. 1. Параллельная декомпозиция задачи

точное количество простых вычислителей (CUDA-ядер) для покрытия требований метода дифференциальной эволюции по размеру кластера векторов [8]. В результате на графическом процессоре может одновременно обрабатываться количество лигандов как минимум равное количеству имеющихся мультипроцессоров.

Метод дифференциальной эволюции для отдельного лиганда декомпозируется за счёт одновременной обработки нескольких векторов из кластера. Таким образом, один вектор с приближенным решением обрабатывается отдельной нитью графического процессора на отдельном CUDA-ядре. Алгоритм одной итерации метода дифференциальной эволюции для одной нити графического процессора представлен на рисунке 2.

Начальная инициализация векторов кластера происходит случайным образом по углам вращения всего лиганда (в пределах 360 градусов) и по торсионным углам (в пределах 10 градусов). Смещения в кластере устанавливаются таким образом, чтобы равномерно распределить векторы по сетке силового поля. Трансформация лиганда включает: вращение всего лиганда вокруг координатных осей, вращение каждого торсионного угла, смещение всего лиганда по трём измерениям. Величина соответствующих изменений определяется тестовым вектором.

При оценке энергии выполняется проверка на выход лиганда за границы сетки силового поля. Кроме того, выполняется проверка на внутренние столкновения атомов лиганда, что может происходить при вращении торсионных углов.

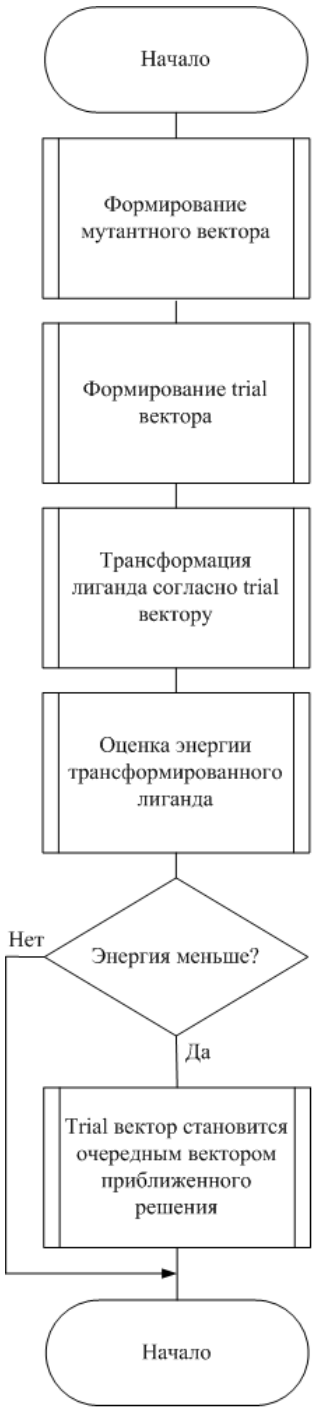

Рис. 2. Блок-схема алгоритма итерации метода дифференциальной эволюции

Для выполнения проверки на этапе предобработки лиганда формируется двумерная clash-карта размерностью N на N, где N – количество атомов в лиганде. При этом ячейка на пересечении строки i и столбца j определяет необходимость проверки дистанции между парой атомов i-j, а также требуемую минимальную дистанцию. Такая организация позволяет, в случае необходимости, гибко настраивать проверяемую дистанцию, а также обеспечивает группировку запросов к памяти графического процессора, так как проверка обязательна для каждого CUDA-ядра и выполняется последовательно для каждого атома.

В случае если энергия лиганда, трансформированного согласно тестовому вектору, меньше энергии, вычисленной на предыдущей итерации, тестовый вектор становится новым приближенным решением.

4. Результаты

Испытания предложенного решения производились на компьютере, укомплектованном двумя процессорами Intel Xeon E5-2640 2.5GHz (12 ядер; 24 вычислительных потока в режиме Hyper Threading), 256 ГБ оперативной памяти и четырьмя графическими процессорами NVIDIA Kepler K20m. Также дополнительные испытания проводились на графических процессорах NVIDIA Kepler K40m.

В качестве альтернатив было выбрано два наиболее распространенных программных пакета для выполнения молекулярного докинга: AutoDock4 и AutoDock Vina [2, 10, 11]. AutoDock Vina запускалась несколько раз для различных конфигураций (6,12 и 24 потока). Предлагаемая реализация обозначена как GDock.

В первой серии тестов были произведены замеры производительности в расчете на один лиганд (таблица 1). Время подготовки сеток силовых полей для AutoDock4 программой AutoGrid4 не включено в результат. Время обработки одного комплекса на графическом процессоре вычислено как время выполнения порции лигандов (размер порции равен количеству мультипроцессоров процессоров), делённое на количество мультипроцессоров. В качестве биомишени была выбрана молекула PDB:1ULB с лигандом ZINC: 895129.

Таблица 1. Время выполнения молекулярного докинга в расчете на один лиганд

Программа	**Время выполнения, с**
AutoDock Vina (cpu=24)	16
AutoDock Vina (cpu=12)	15.9
AutoDock Vina (cpu=6)	19.5
AutoDock4	153.2
GDock NVIDIA Tesla K20m	3.12
GDock NVIDIA Tesla K40m	1.27

При проведении тестов в режиме скрининга (обработки большого количества лигандов) AutoDock 4 был исключён, из-за низкой скорости работы (при этом следует отметить, что AutoDock4 продемонстрировал максимальную точность позиционирования лиганда). Для выполнения скрининга была выбрана биомишень PDB:3PTB с лигандом ZINC:36634. К тестовому набору было случайным образом добавлено 99 лигандов, содержащихся в базе данных химических соединений разрешённых к использованию лекарств. При этом для предлагаемой реализации дополнительно производились замеры при использовании нескольких графических процессоров, установленных в том же компьютере. Полученные результаты представлены в таблице 2.

Таблица 2. Время выполнения молекулярного докинга для набора из 100 лигандов

Программа	**Время выполнения, с**
AutoDock Vina (cpu=24)	2253
AutoDock Vina (cpu=12)	2239
AutoDock Vina (cpu=6)	2302
GDock NVIDIA Tesla K20m	322
GDock NVIDIA Tesla K20m X3	85
GDock NVIDIA Tesla K40m	131
GDock NVIDIA Tesla K40m X3	80

С целью оценки точности позиционирования для набора пар биомишень – лиганд было рассчитано среднеквадратичное отклонение (RMSD) предсказанного положения лиганда от истинного положения, которое достаточно точно оценено средствами кристаллографии. Для тестов были выбраны структуры с PDB идентификаторами 1ULB, 2MCP, 3PTB вместе с соответствующими лигандами. Результаты тестов представлены в таблице 3.

Трёхмерные структуры и результаты их позиционирования представлены на рисунке 3.

Таблица 3. Среднеквадратичное отклонение предсказанных конформаций молекулы биомишени и лиганда от истинного значения.

Комплекс	**RMSD,Ангстрем**
PDB:1ULB	2.15
PDB:2MCP	1.79
PDB:3PTB	0.44

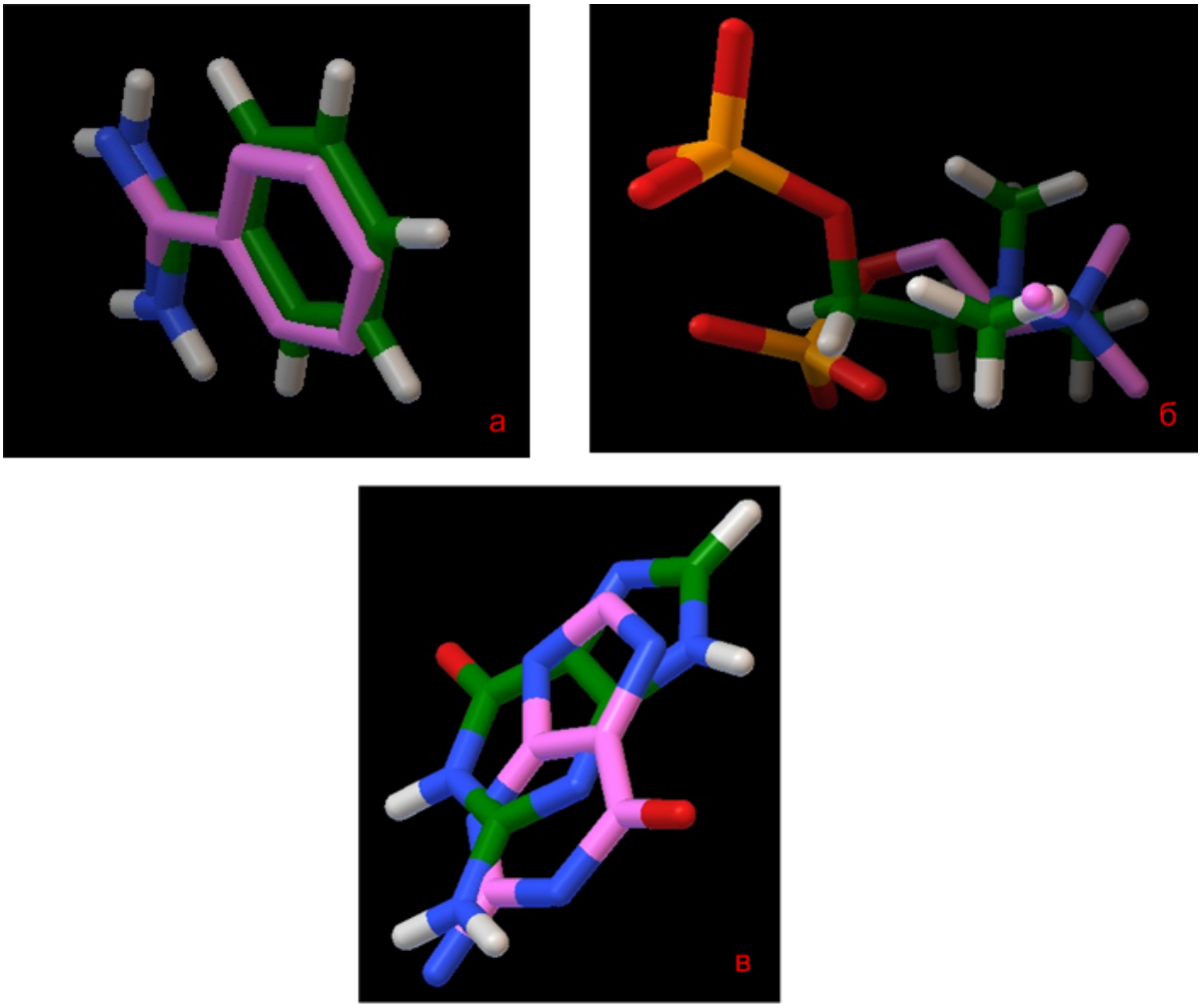

Рис. 3. Результаты предсказания трёхмерных структур а) 3PTB б) 2MCP в) 1ULB

Таким образом, в ходе работы проанализированы существующие методы оптимизации и их реализации для решения задачи молекулярного лиганд-белкового докинга. Выбран метод оптимизации, подходящий для графических процессоров, предложена его реализация, выполнена его реализация. Предлагаемое решение демонстрирует приемлемый уровень точности позиционирования лигандов, несмотря на то, что применяется только глобальная оптимизация. При этом программа де-

монстрирует существенный прирост производительности по сравнению с существующими решениями: в 17 раз при использовании одного графического процессора и в 28 раз при использовании трёх графических процессоров.

Список литературы

1. Фарков М.А. Вычисление сеток взаимодействия молекул с использованием графических процессоров // Исследования наукограда. 2013. 3(5), июль–сентябрь. С. 49–52.

2. Trott O., Olson A. AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading // Journal Computational Chemistry. 2010. 31. P. 455–461.

3. Jones G., Willett P., Glen R., Leach A., Taylor R. Development and Validation of a Genetic Algorithm for Flexible Docking // Journal of Molecular Biology. 1997. 267. P. 727–748.

4. Stroganov O.V., Novikov F., Stroylov V., Kulkov V., Chilov G. Lead finder: an approach to improve accuracy of protein-ligand docking, binding energy estimation, and virtual screening // Journal of Chemical Information and Modeling. 2008. 48(12). P. 2371–2385.

5. Liu M., Wang S. MCDOCK: A Monte Carlo simulation approach to the molecular docking problem // Journal of Computer-Aided Molecular Design. 1999. 13. P. 435–451.

6. Meiler J., Baker D. ROSETTALIGAND: Protein–Small Molecule Docking with Full Side-Chain Flexibility // PROTEINS: Structure, Function, and Bioinformatics. 2006. 65. P. 538–548.

7. Korb O., Stutzle T., Exner T.E. Accelerating molecular docking calculations using graphics processing units // Journal of chemical information and modeling. American Chemical Society. 2011. 51(4). P. 865–876.

8. Storn R., Price K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces // Journal of Global Optimization. 1997. 11(4). P. 341–359.

9. Pechan I., Feher B. Hardware Accelerated Molecular Docking: A Survey // Bioinformatics. ISBN 978-953-51-0878-8. Available from: http://www.intechopen.com/books/bioinformatics/hardware-accelerated-molecular-docking-a-survey

10. Morris G.M., Huey R., Lindstrom W., Sanner M.F., Belew R.K., Goodsel, D.S., Olson, A.J. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility // Journal of Computational Chemistry. 2009. 30. P. 2785–2791.

11. Morris G.M., Goodsell D.S., Halliday R.S., Huey R., Hart W.E., Belew R.K., Olson A.J. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function // Journal of Computational Chemistry. 1998. 19. P. 1639–1662.

12. Farkov M.A. Calculation of force field grids for molecular docking using GPU // Journal of Siberian Federal University. Biology. 2014. 7(1). P. 4–13.

Application of Numerical Optimization Methods to Perform Molecular Docking on Graphics Processing Units

Farkov M.A., Legalov A.I.

Siberian Federal University
Svobodny pr., Krasnoyarsk, 660041, Russia.

Keywords: GPGPU, CUDA, molecular docking, virtual screening, ligand-protein docking

An analysis of numerical optimization methods for solving a problem of molecular docking has been performed. Some additional requirements for optimization methods according to GPU architecture features were specified. A promising method for implementation on GPU was selected. Its implementation was described and performance and accuracy tests were performed.

Сведения об авторах:
Фарков Михаил Александрович,
Сибирский федеральный университет, аспирант,
Легалов Александр Иванович,
Сибирский федеральный университет,
профессор, заведующий кафедрой вычислительной техники.